Project Introduction and Project Purpose

"https://www.teamrankings.com/nba/stats" website contains statistics of past NBA games. The purpose of the project is to use the NBA statistics of the Chicago team to change the win rate of the team by 3 classes (win rate increases, win rate is fixed, win rate decreases) is to train predictive machine learning models.

Question 1

Dataset

Our data set is for the Chicago team in the matches; It consists of 8 features: ball blocking rate, defensive play effectiveness, offensive play effectiveness, steal rate, total rebounds per game, rate of shots hitting the hoop, turnover (counter attack) percentage, win rate change.

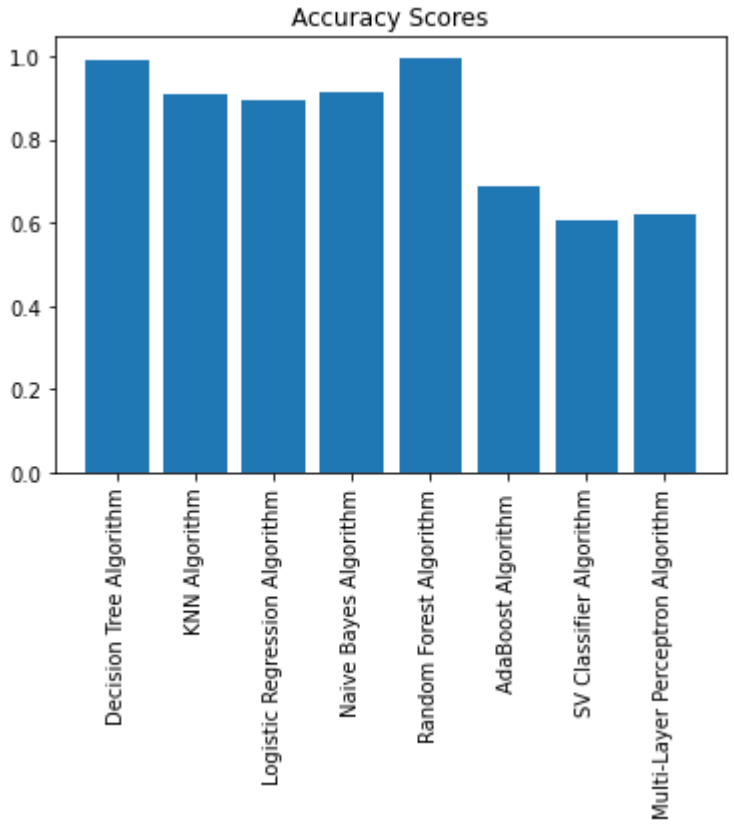
Data Gathering Process

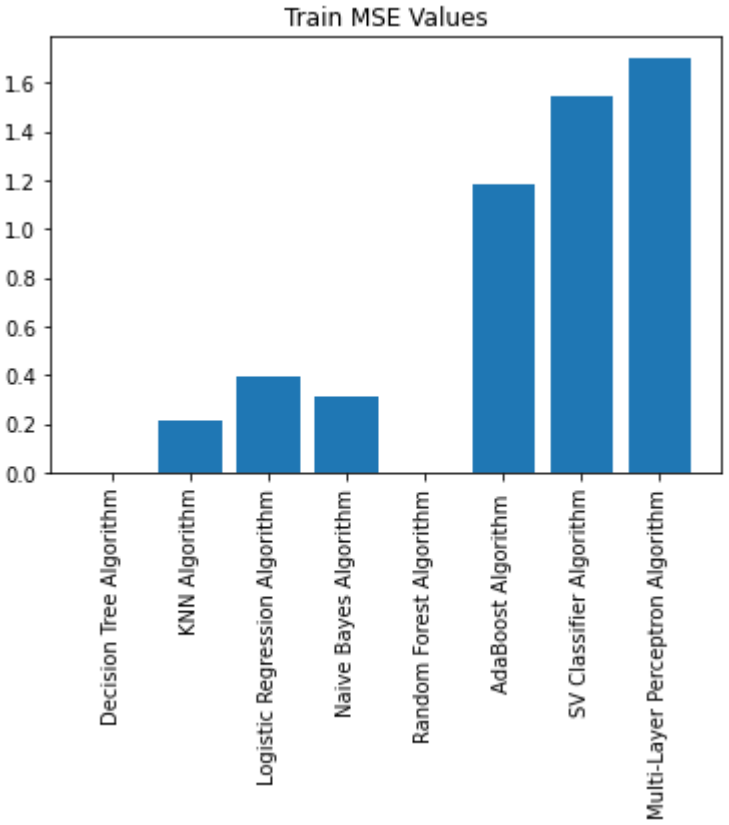
We scraped the data from the website “https://www.teamrankings.com/nba/stats” with the help of scraper bot from 2013 to June 2021 (approximately 3000 data). We used the 8 most effective features out of 136 features to analyze the NBA matches on the "teamrankings.com" site. We got help from several articles and studies for feature extraction [1].

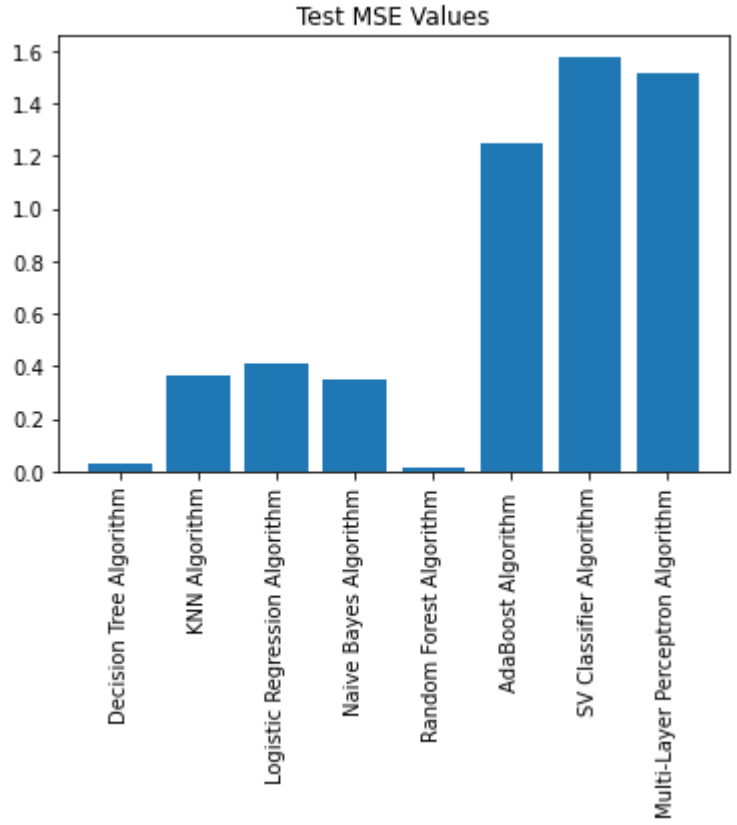
Question 3

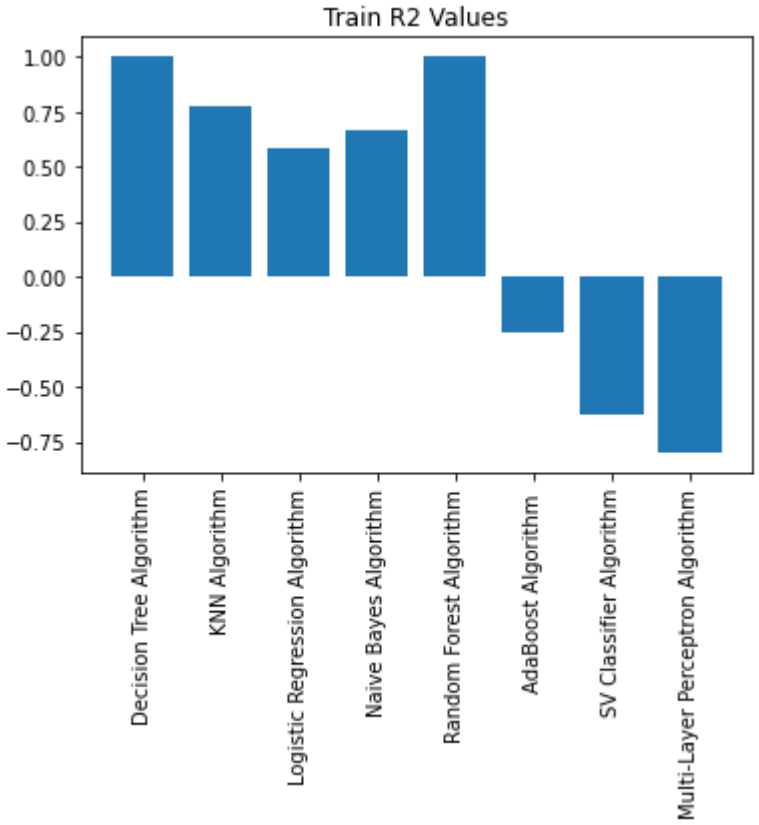
Training Models

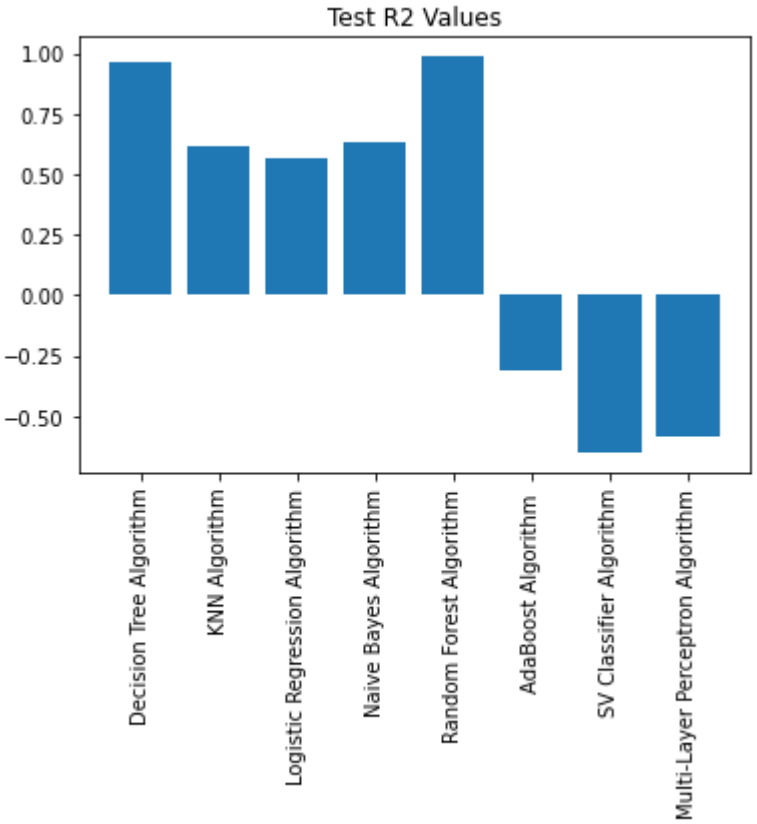
We use Decision Tree, KNN, Logistic regression, Naïve Bayes, Random Forest, AdaBoost, SV classifier and Multi-Layer Perceptron (MLP/ANN) algorithms to training.




Train MSE Values
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Decision Tree Algorithm
KNN Algorithm
Logistic Regression Algorithm
Naive Bayes Algorithm
Random Forest Algorithm
AdaBoost Algorithm
SV Classifier Algorithm
Multi-Layer Perceptron Algorithm


Test MSE Values
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
Decision Tree Algorithm
KNN Algorithm
Logistic Regression Algorithm
Naive Bayes Algorithm
Random Forest Algorithm
AdaBoost Algorithm
SV Classifier Algorithm
Multi-Layer Perceptron Algorithm


Train R2 Values
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
Decision Tree Algorithm
KNN Algorithm
Logistic Regression Algorithm
Naive Bayes Algorithm
Random Forest Algorithm
AdaBoost Algorithm
SV Classifier Algorithm
Multi-Layer Perceptron Algorithm


Test R2 Values
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
Decision Tree Algorithm
KNN Algorithm
Logistic Regression Algorithm
Naive Bayes Algorithm
Random Forest Algorithm
AdaBoost Algorithm
SV Classifier Algorithm
Multi-Layer Perceptron Algorithm

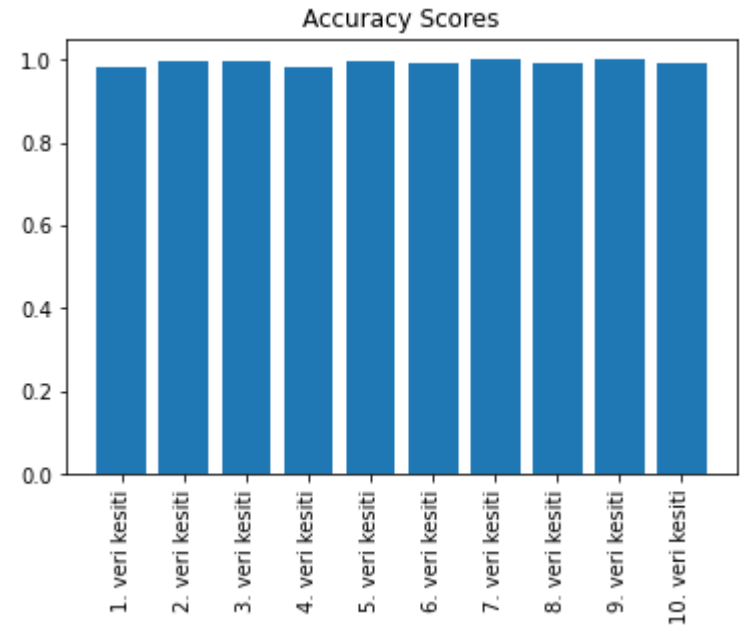
Q3-Results

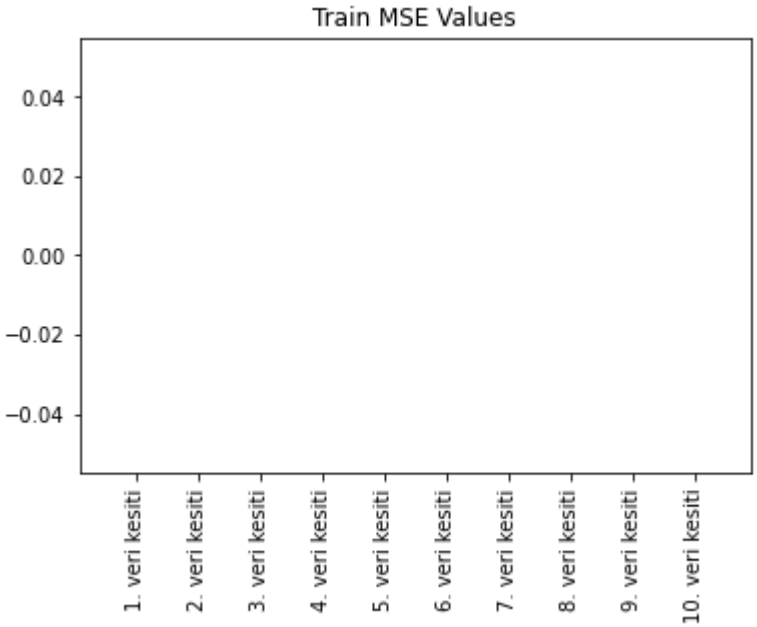
The results we obtained after the algorithm training are as in the graphics. According to the graphs, the model with the best results on our dataset: Random Forest, the model with the worst result: SV Classifier.

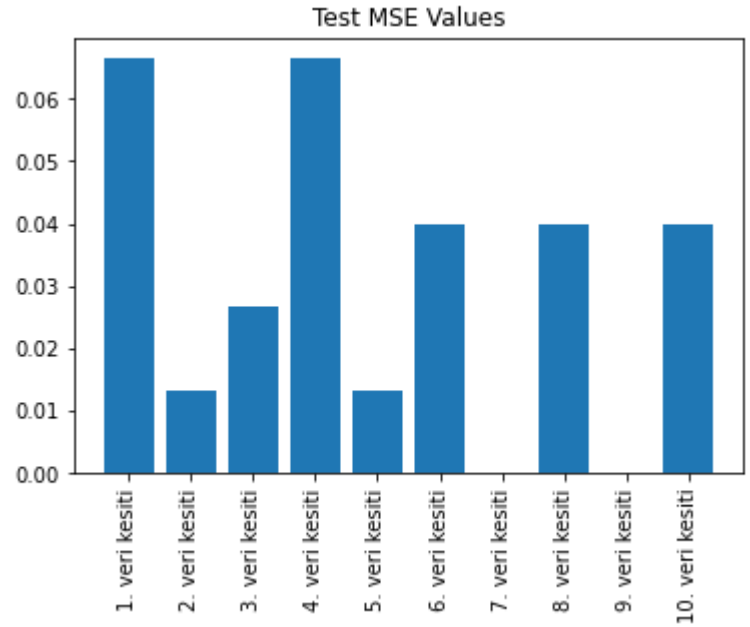
Question 4

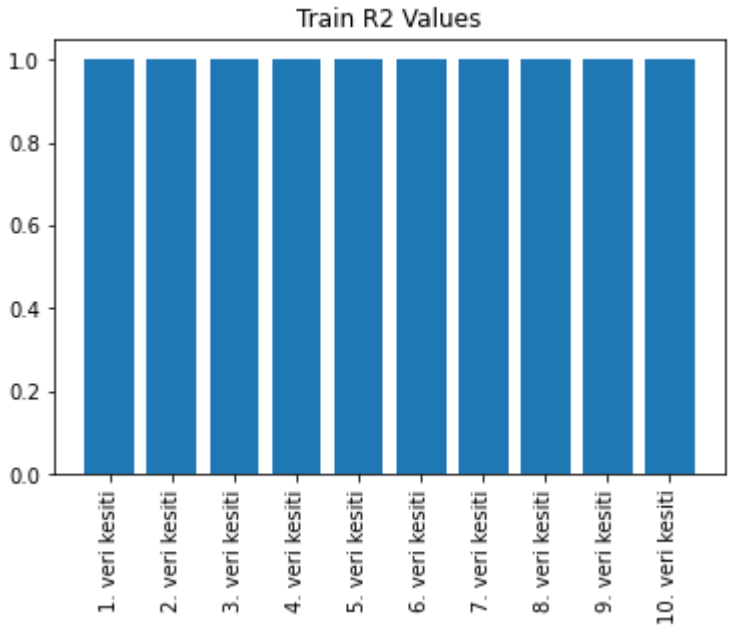
Training Models with 10-Fold Cross Validation

Decision Tree:

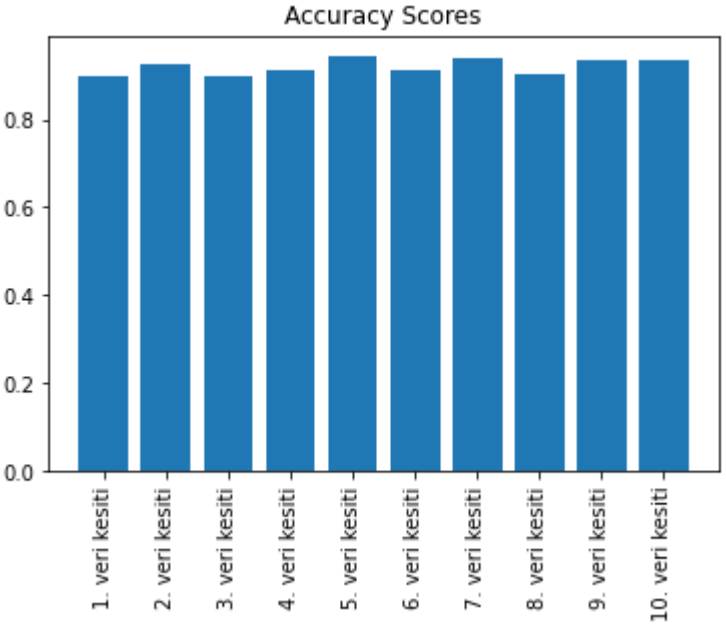


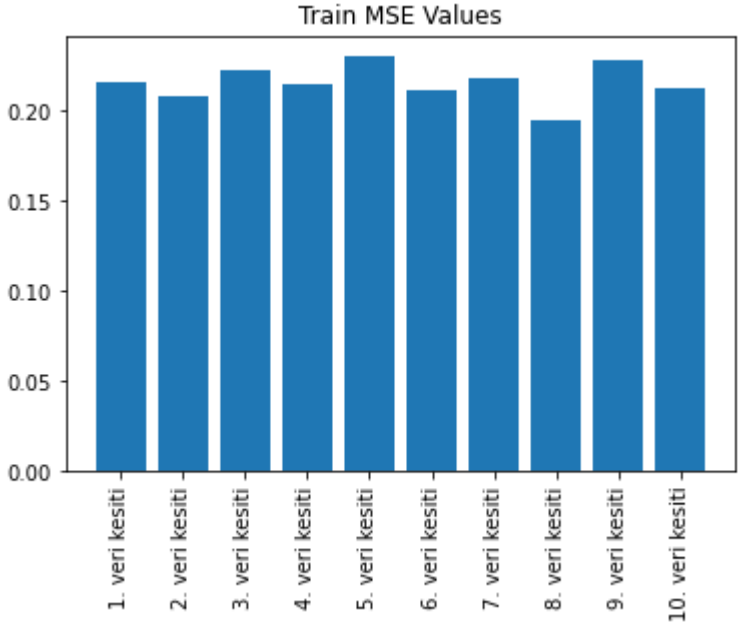


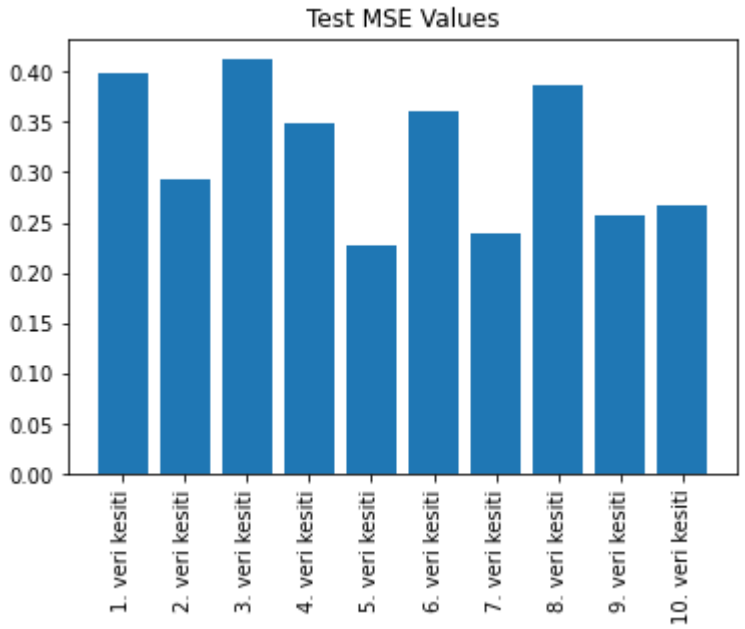


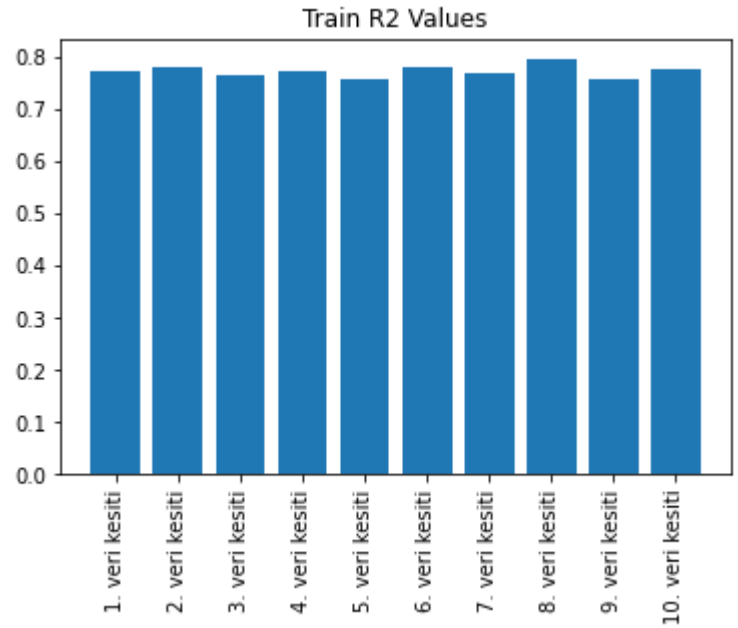


KNN Algorithm:

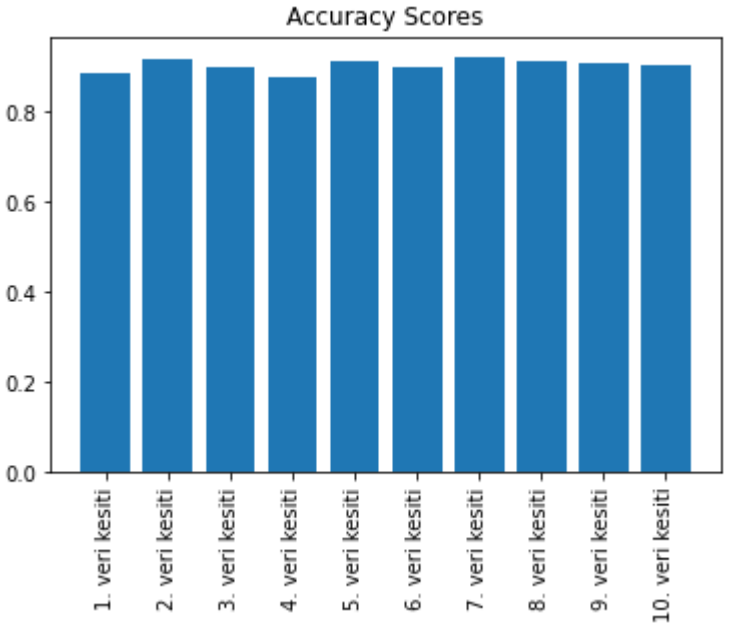

Accuracy Scores
0.0
0.2
0.4
0.6
0.8
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti


Train MSE Values
0.00
0.05
0.10
0.15
0.20
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

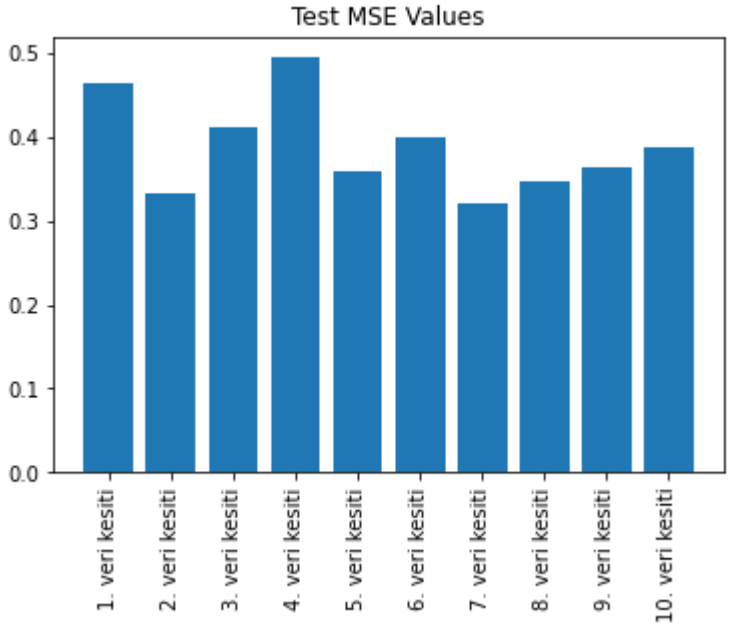


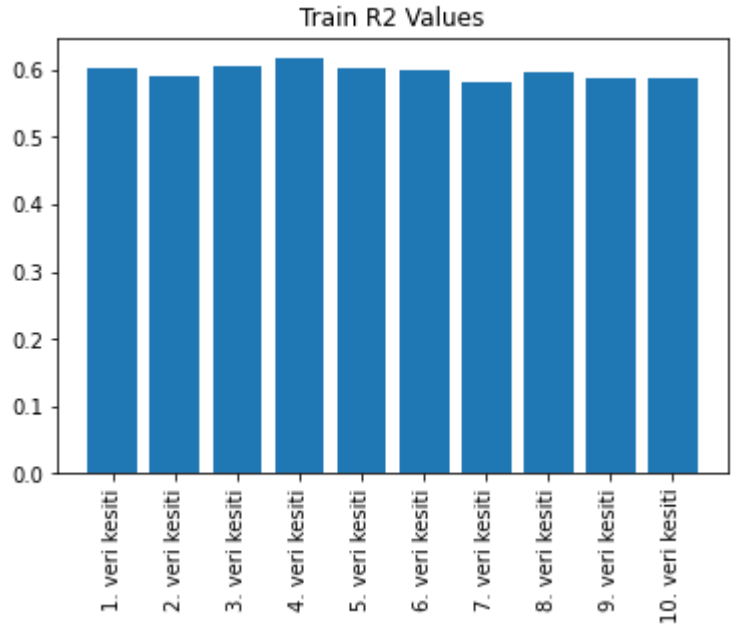


Logistic Regression

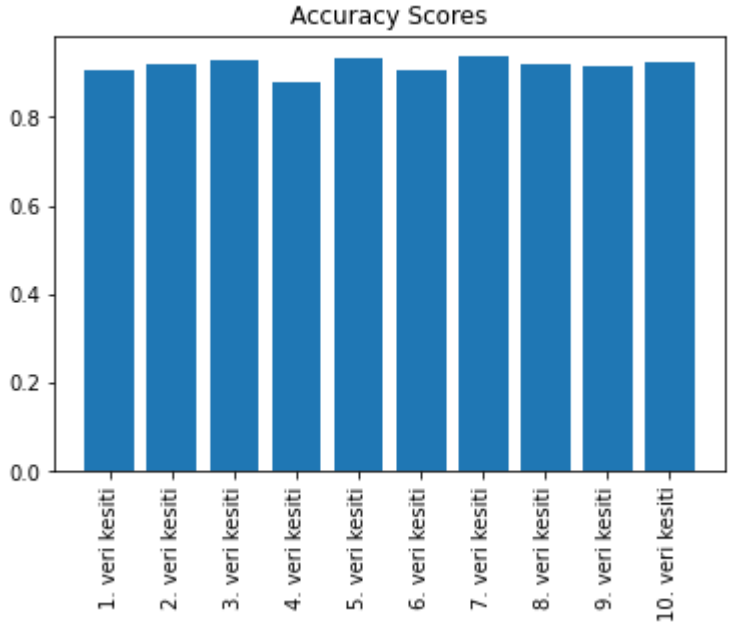

Accuracy Scores
0.0
0.2
0.4
0.6
0.8
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti


Train MSE Values
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
0.40
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

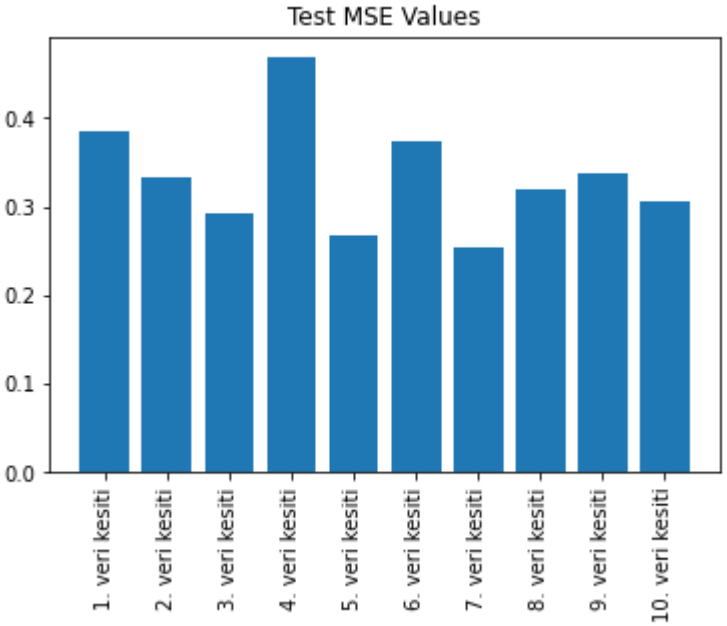


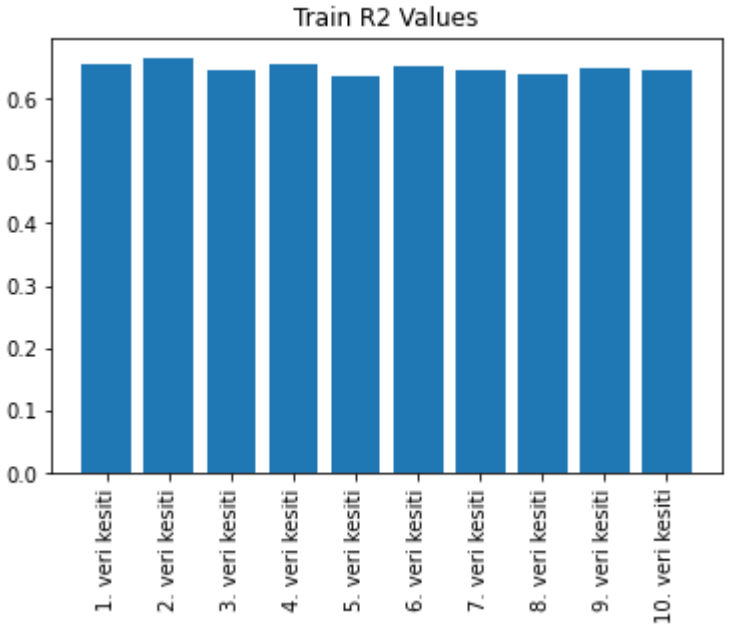


Naive Bayes Algorithm:

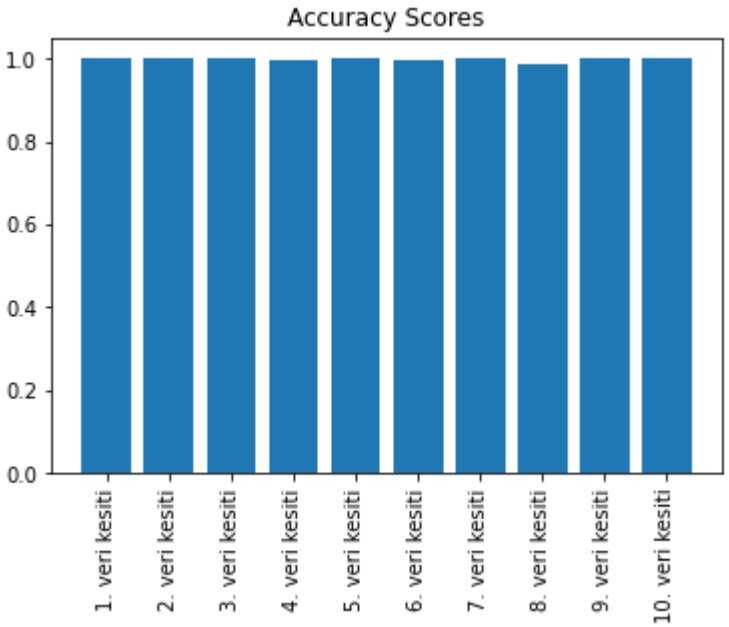

Accuracy Scores
0.0
0.2
0.4
0.6
0.8
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

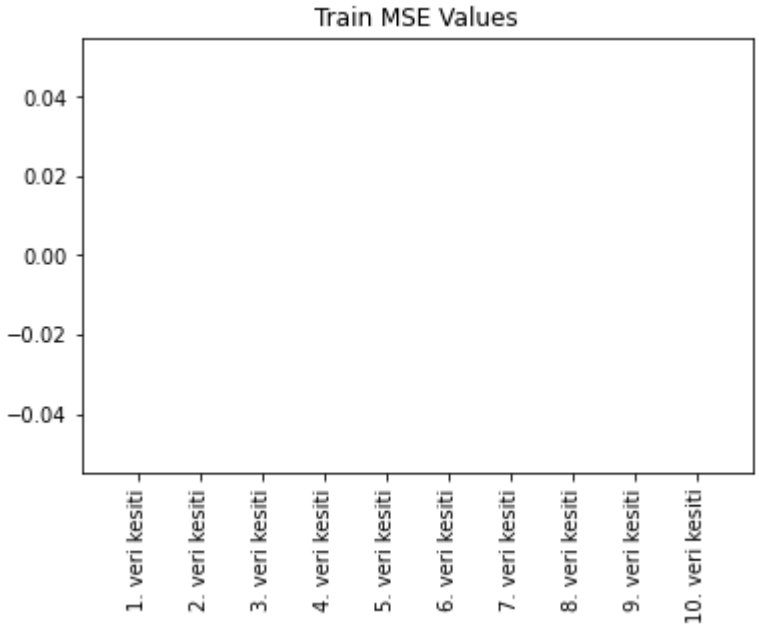

Train MSE Values
0.00
0.05
0.10
0.15
0.20
0.25
0.30
0.35
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

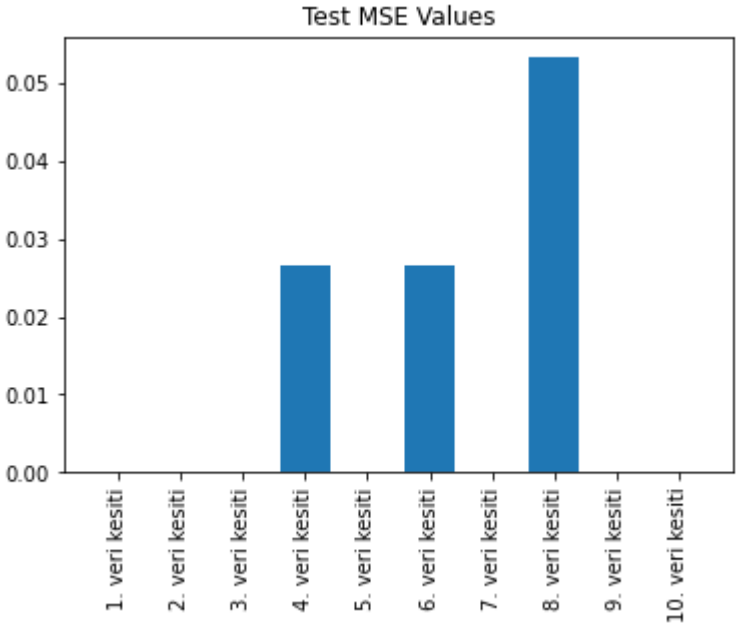


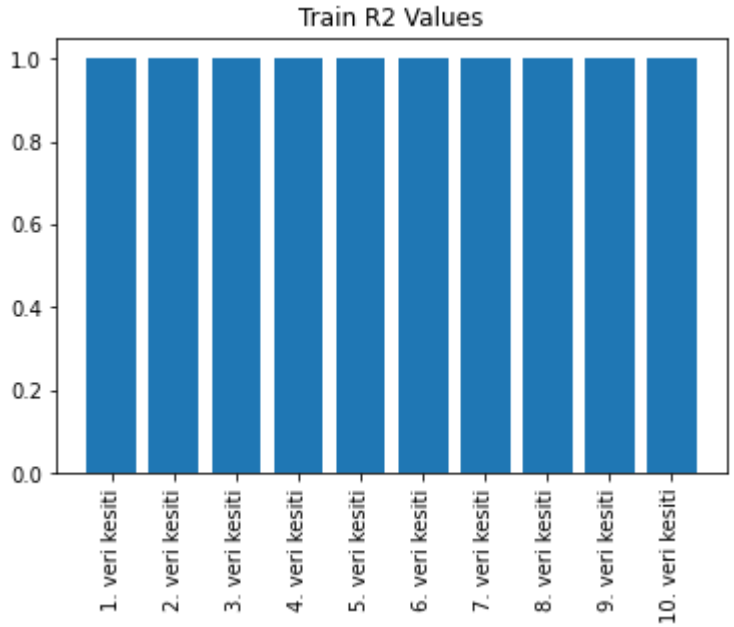


Random Forest Algorithm:

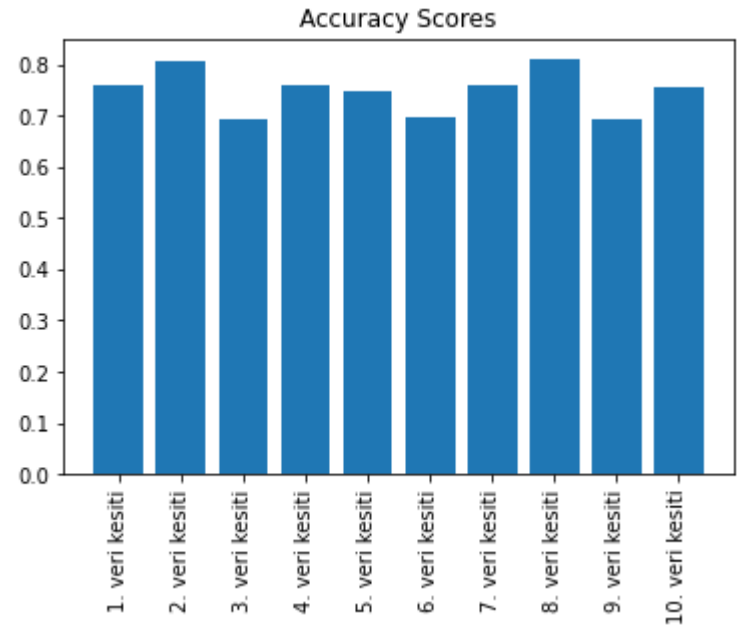

Accuracy Scores
1.0
0.8
0.6
0.4
0.2
0.0
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

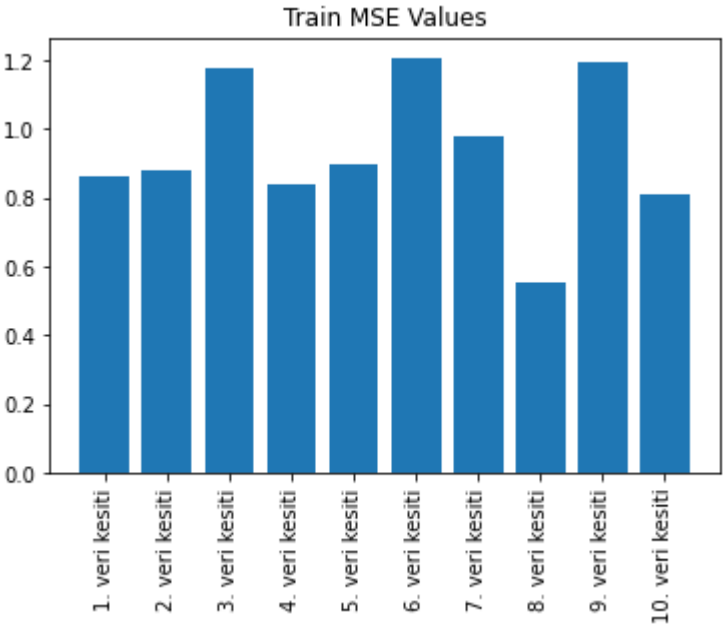

Train MSE Values
0.04
0.02
0.00
−0.02
−0.04
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

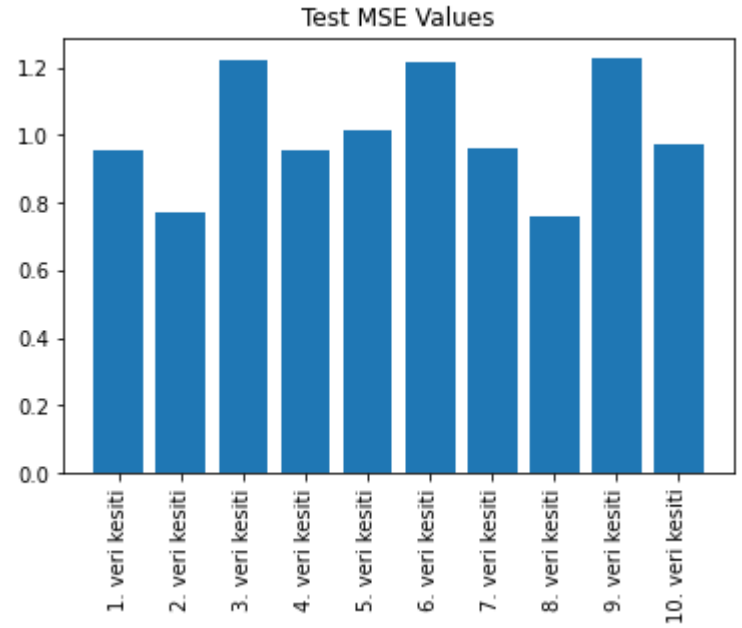


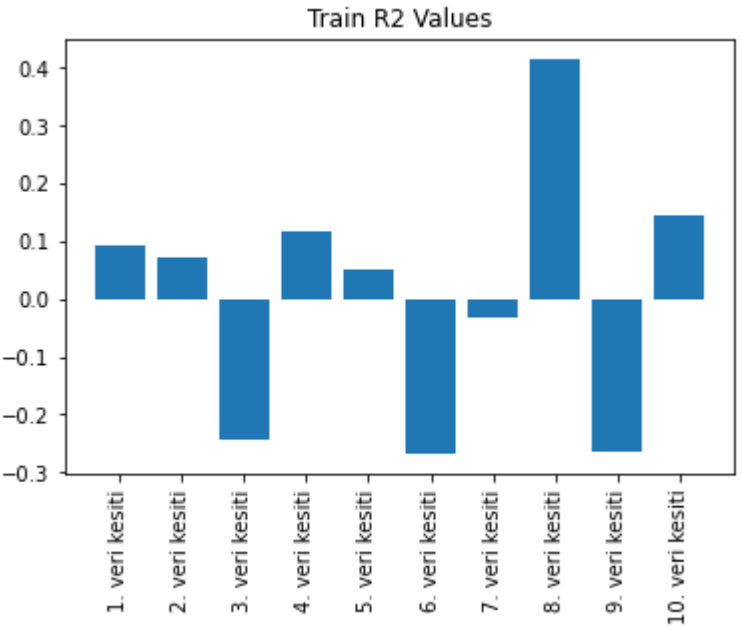


AdaBoost Algorithm:

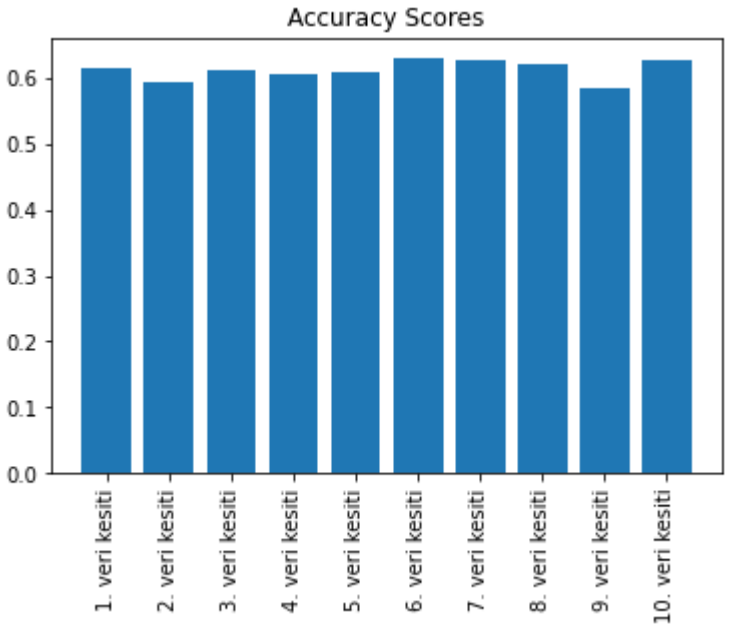

Accuracy Scores
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

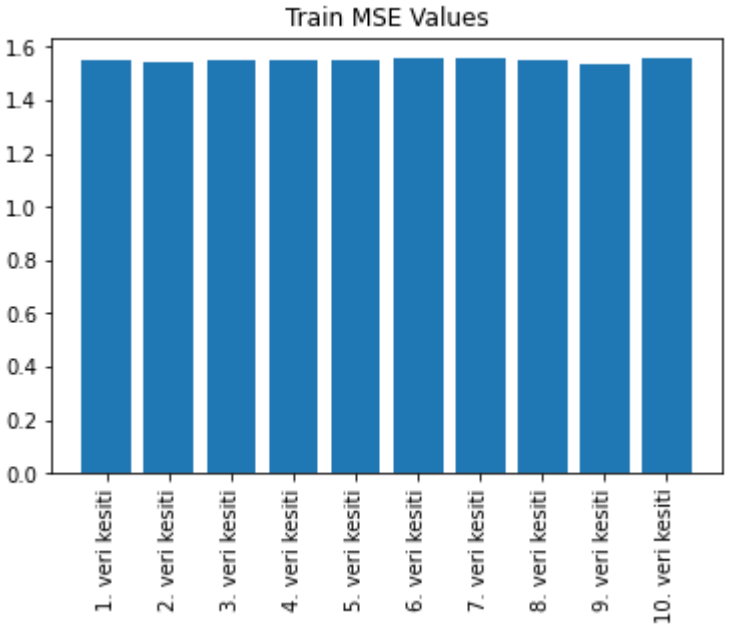

Train MSE Values
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

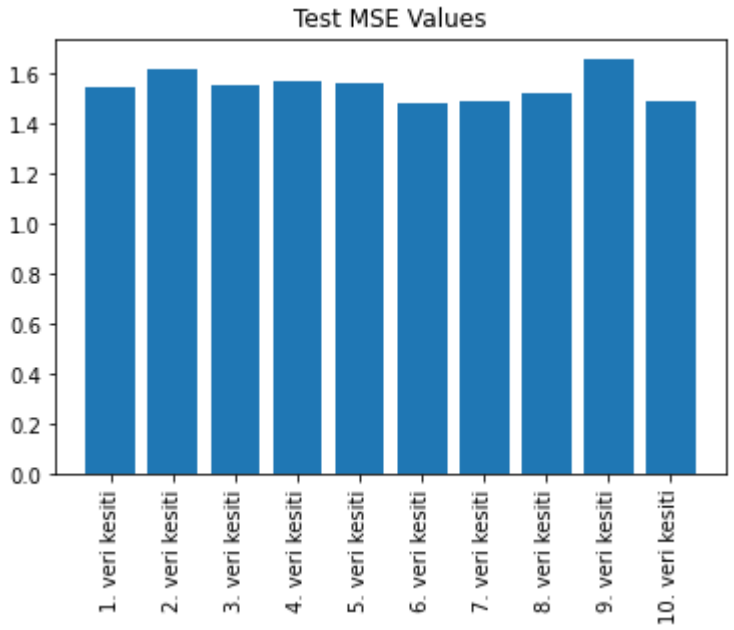


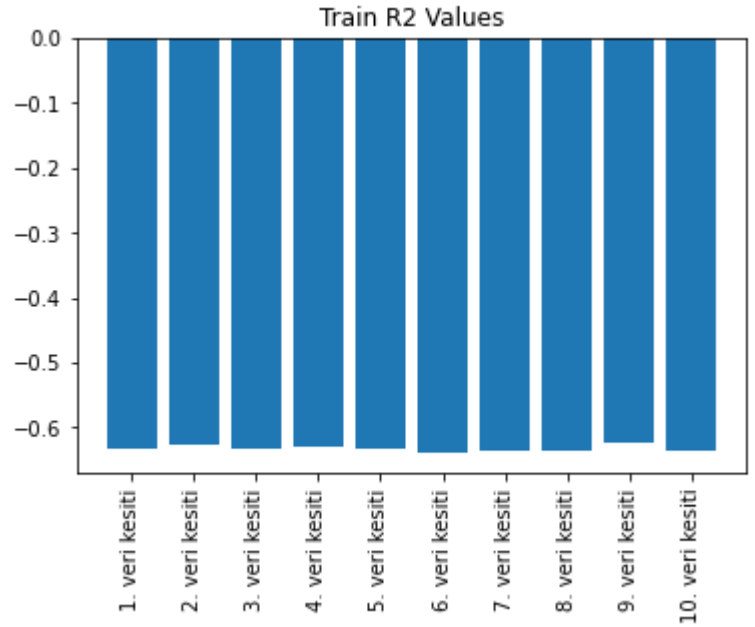


SV Classifier Algorithm:

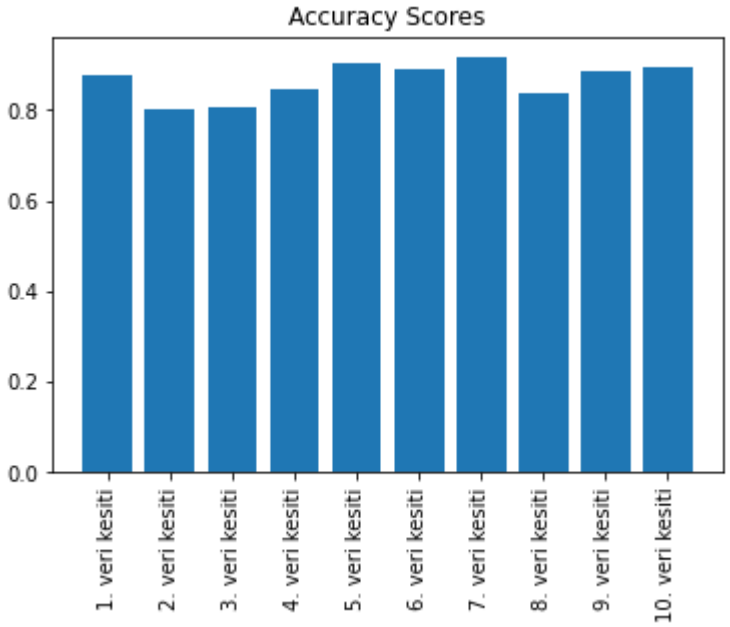

Accuracy Scores
0.0
0.1
0.2
0.3
0.4
0.5
0.6
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

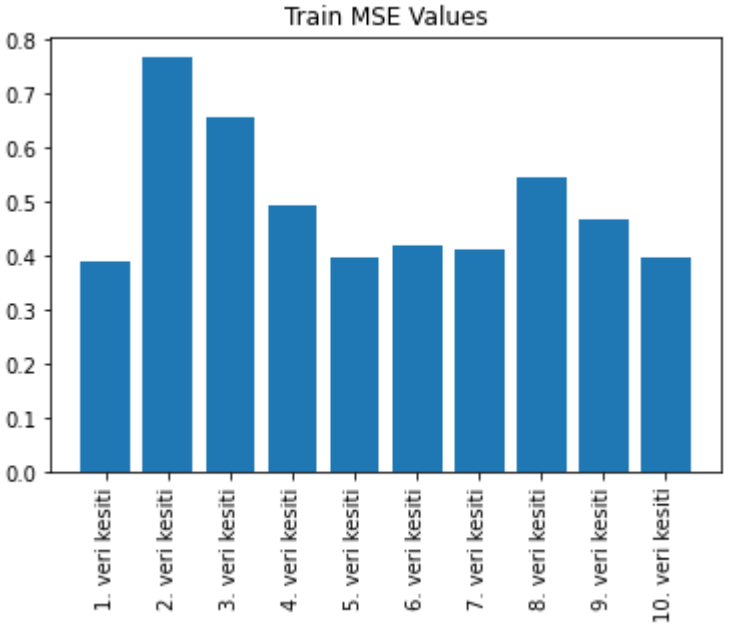

Train MSE Values
0.0
0.2
0.4
0.6
0.8
1.0
1.2
1.4
1.6
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti

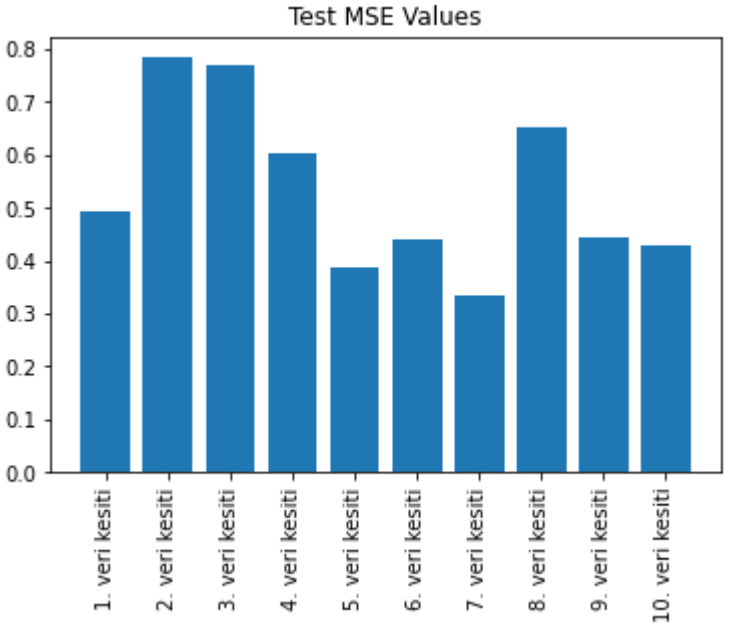


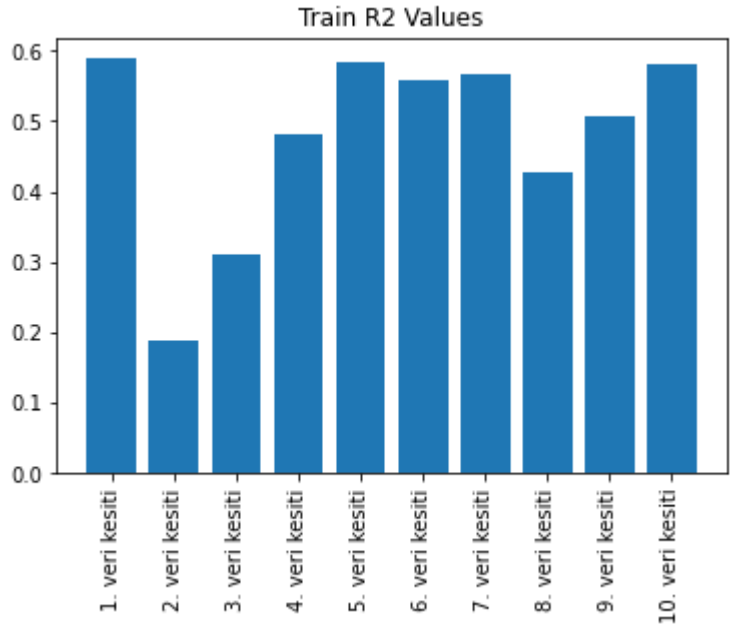


Multi-Layer Perceptron Algorithm:


Accuracy Scores
0.0
0.2
0.4
0.6
0.8
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti


Train MSE Values
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
1. veri kesiti
2. veri kesiti
3. veri kesiti
4. veri kesiti
5. veri kesiti
6. veri kesiti
7. veri kesiti
8. veri kesiti
9. veri kesiti
10. veri kesiti





Q4-Results

The data of the algorithms after the training are as in the graphs. As can be seen from the graphs, the best models are Random Forest Classifier and Decision Tree Model.

Question 5-6

Hyperparameters:

We optimized the RandomForestClassifier model, where we got the best results, according to its hyperparameters. The hyperparameters we base on are as follows:

```
"n_estimators": [20, 40, 60, 80, 100],
"criterion": ["gini", "entropy"],
"max_depth": [2, 4, 6],
"min_samples_split": [3, 6, 9],
"min_samples_leaf": [2, 4, 6],
"max_features": ["auto", "sqrt", "log2"]
```

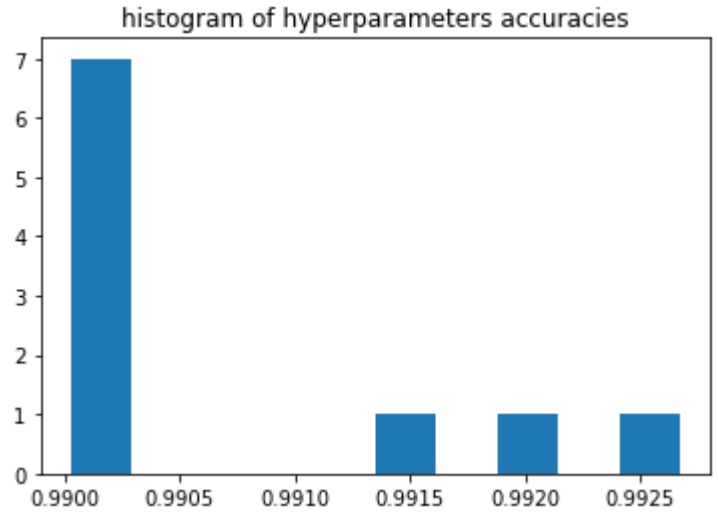
We used the Grid Search method to optimize the hyperparameters. The top 3 hyperparameter combinations are as follows:

0.9926754385964914 accuracy
'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 9, 'n_estimators': 80

0.9920175438596492 accuracy
'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 100

0.9900350877192983 accuracy
'criterion': 'gini', 'max_depth': 6, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 80


hyperparameters accuracies
1.0
0.8
0.6
0.4
0.2
0.0
{'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 9, 'n_estimators': 80}
{'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 9, 'n_estimators': 80}
{'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 100}
{'criterion': 'gini', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 60}
{'criterion': 'gini', 'max_depth': 6, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 80}
{'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 4, 'min_samples_split': 6, 'n_estimators': 60}
{'criterion': 'entropy', 'max_depth': 6, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 6, 'n_estimators': 40}
{'criterion': 'gini', 'max_depth': 6, 'max_features': 'log2', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 100}
{'criterion': 'entropy', 'max_depth': 6, 'max_features': 'sqrt', 'min_samples_leaf': 2, 'min_samples_split': 6, 'n_estimators': 80}
{'criterion': 'entropy', 'max_depth': 6, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 3, 'n_estimators': 40}


histogram of hyperparameters accuracies
7
6
5
4
3
2
1
0
0.9900
0.9905
0.9910
0.9915
0.9920
0.9925

Question 7

Predictions:

Since our independent variables in our dataset are a match statistic, we would have to enter an unplayed match statistic to make predictions for the models. In short, we should have predicted the parameters of the match to be played. Therefore, we determined the parameters that we would predict to the independent model based on the averages of the Chicago team's previous game performances. We created "makePrediction" function to make this prediction.

function: makePrediction

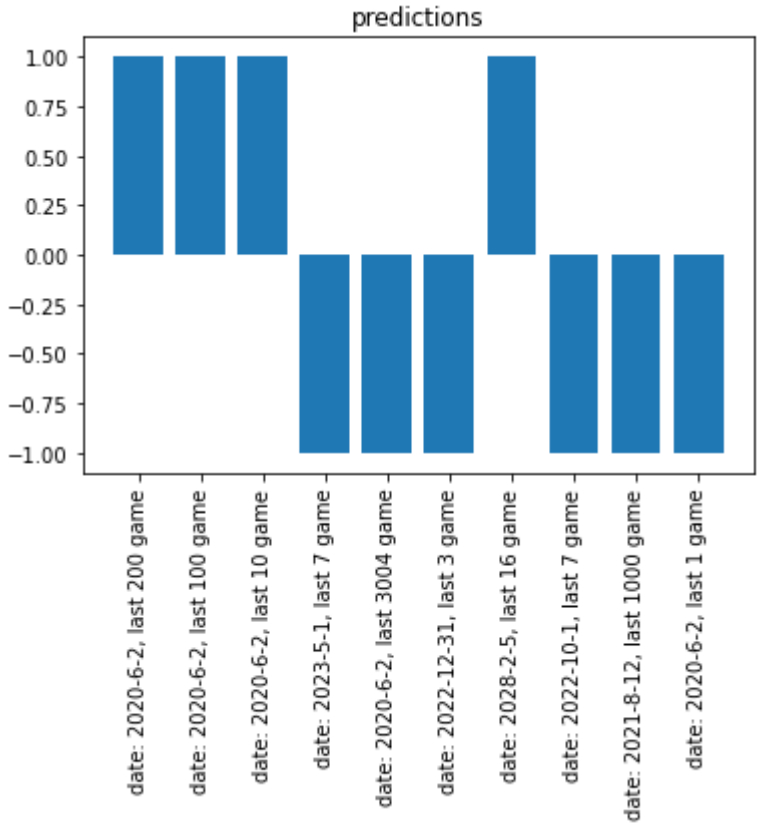
input:

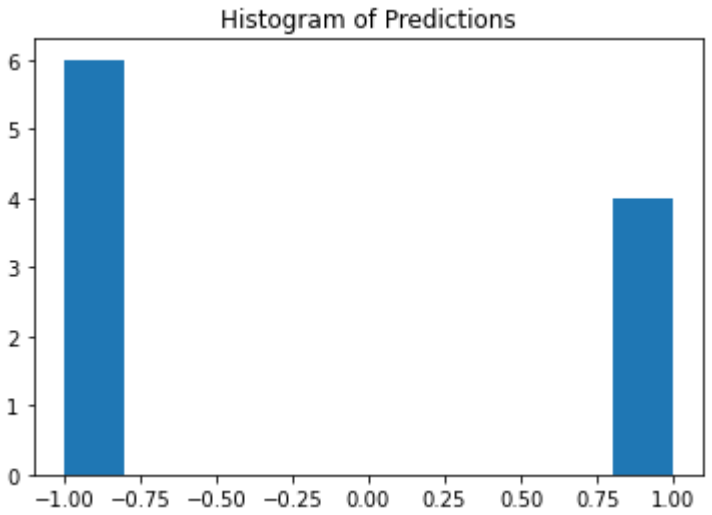
lastGameCount (int): default = rows of data
year (int): default = 2020
month (int): default = 6
day (int): default = 2
model (sklearn.model): default = sklearn.ensemble._forest.RandomForestClassifier

output:

enteredSampleInfo (string)
prediction (int)

The “lastGameCount” parameter in the makePrediction function determines how many last games of the Chicago team will be based on the average of their statistics. By taking the average of the statistics, it allows our model to estimate over them.





References

[1]:https://github.com/JakeKandell/NBA-Predict

https://watchstadium.com/which-nba-statistics-actually-translate-to-wins-07-13-2019/